Experiment No 1: To implement stack ADT using arrays

Aim: To implement stack ADT using arrays

Objective:

1) Understand the use of stack
2) Understand how to import an ADT in an application program
3) Understand the instantiation of stack ADT in an application program
4) Understand how the member function of an ADT are accessed in an application program

Theory: A stack is a list in which all insertions and deletions are made at one end called the top. It is a collection of contiguous cells, stacked on top of each other. The last element to be inserted into the stack will be the first to be removed. Thus stacks are sometimes referred to as Last In First Out (LIFO) lists. The operations that can be performed on a stack are push, pop which are main operations while auxiliary operations are peek, isEmpty and isFull. Push is to insert an element at the top of the stack. Pop is deleting an element that is at the top most position in the stack. Peck simply examines and returns the top most value in the stack without deleting it.Push on an already filled stack and pop on an empty stack results in serious errors so isEmpty and isFull function checks for stack empty and stack full respectively. Before any insertion, the value of the variable top is initialized to -1.

Algorithm:

- Check if the top is equal to -1. If true, then print "Stack is Empty" and return.
- If false, then store the top item in a variable. ● Decrement the top.
- Return the stored item through the variable.

Code:

```
#include<stdio.h>

#include<stdlib.h>

#include<conio.h> int

stack[100],top,n,choice,x,i;
```

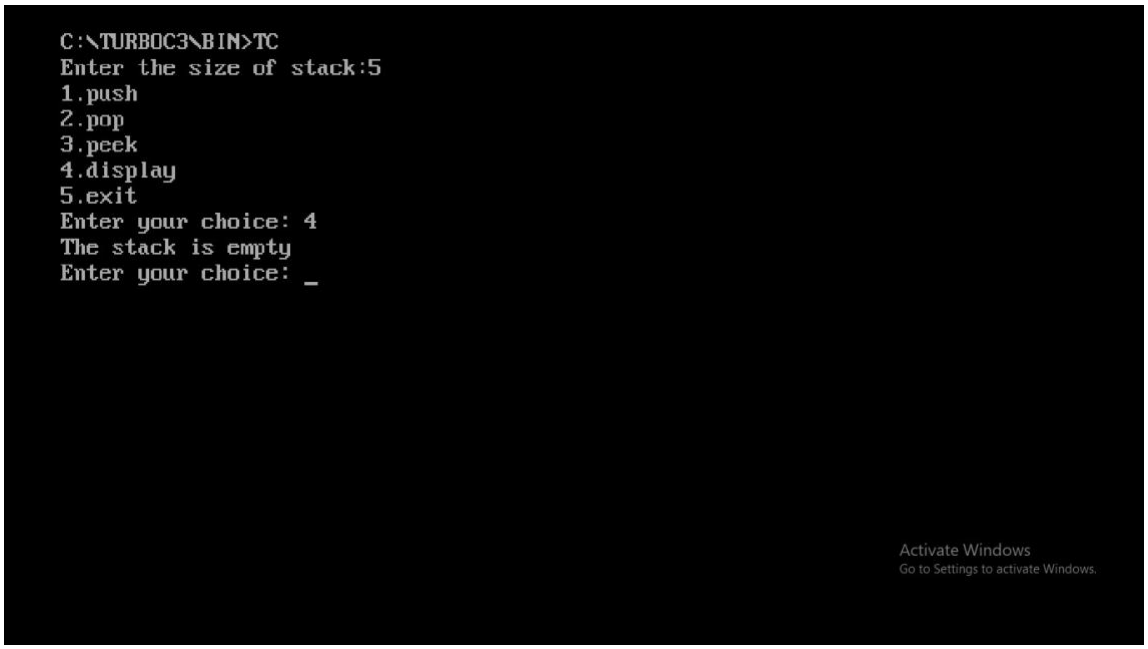
```
void push(); void pop(); void
peek(); void display(); int
main()
{
top=-1; printf("Enter the size of stack:");
scanf("%d",&n);
printf("1.push\n2.pop\n3.peek\n4.display\n5.exit\n");
while(choice!=5)
{
printf("Enter your choice: ");
scanf("%d",&choice); switch
(choice)
{
case 1:
{
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
peek();
break;
}
```

```
case 4:
{
display();
break;
}
case 5:
{ printf("exit
point"); break;
}
default:
{
printf("Enter a valid choice");
}
getch();
return 0;
}
}
}
void push()
{
if (top==n-1)
{
printf("Stack is overflow\n");
}
else
{
printf("Enter a value to be pushed:");
scanf("%d",&x); top++; stack[top]=x;
```

```
}
}
void pop()
{
if (top==n-1)
{
printf("Stack is underflow\n");
}
else
{
printf("Enter a value to be popped:");
scanf("%d",&x); top--; stack[top]=x;
}
}
void peek()
{
printf("The top element of the stack is %d \n",x);
}
void display()
{
if (top==0)
{
printf("The element in the stack are:\n"); for(i=top;i>=0;i--) printf("%d\n",stack[i]);
}
else
{
printf("The stack is empty\n");
}
```

}

Output:



Conclusion :

In summary, stacks can be classified as ADTs, or Abstract Data Types, due to the unchanging nature of operations like push and pop. This essential consistency persists regardless of the implementation's data structure or the programming language utilized, reinforcing the stability of the specifications across various contexts.